



PATENT ABSTRACTS OF JAPAN

(11) Publication number: **09023422 A**(43) Date of publication of application: **21.01.97**

(51) Int. Cl.

H04N 7/24
G06T 9/00
(21) Application number: **07171816**(22) Date of filing: **07.07.95**(71) Applicant: **NIPPON TELEGR & TELEPH
CORP <NTT>**(72) Inventor: **OSAKO FUMINORI
YASHIMA YOSHIYUKI**

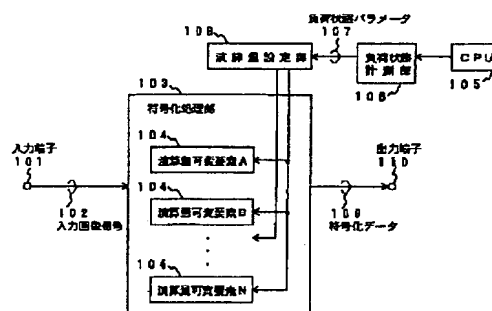
(54) PICTURE ENCODING AND DECODING METHOD

(57) Abstract:

PROBLEM TO BE SOLVED: To provide a method for realizing stable encoding and decoding with high quality in given operation ability in accordance with the change of the operation ability of CPU when a picture signal is encoded and decoded on a real time.

SOLUTION: A load state measuring part 106 measures the load state of CPU 105 and transmits a load state parameter 107 to an operation quantity setting part 108. The operation quantity setting part 108 sets the operation quantity of an encoding processing part 103 in accordance with the load state based on the load state parameter 107. The encoding processing part 103 encoding-processes an input picture signal 102 with a processing algorithm by which the operation quantity of an operation quantity variable element 104 becomes operation quantity which is set and executes encoding.

COPYRIGHT: (C)1997,JPO



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-23422

(43) 公開日 平成9年(1997)1月21日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
H 0 4 N 7/24			H 0 4 N 7/13	Z
G 0 6 T 9/00			G 0 6 F 15/66	3 3 0 A

審査請求 未請求 請求項の数2 O L (全 8 頁)

(21) 出願番号 特願平7-171816

(22) 出願日 平成7年(1995)7月7日

(71) 出願人 000004226

日本電信電話株式会社

東京都新宿区西新宿三丁目19番2号

(72) 発明者 大迫 史典

東京都千代田区内幸町1丁目1番6号 日

本電信電話株式会社内

(72) 発明者 八島 由幸

東京都千代田区内幸町1丁目1番6号 日

本電信電話株式会社内

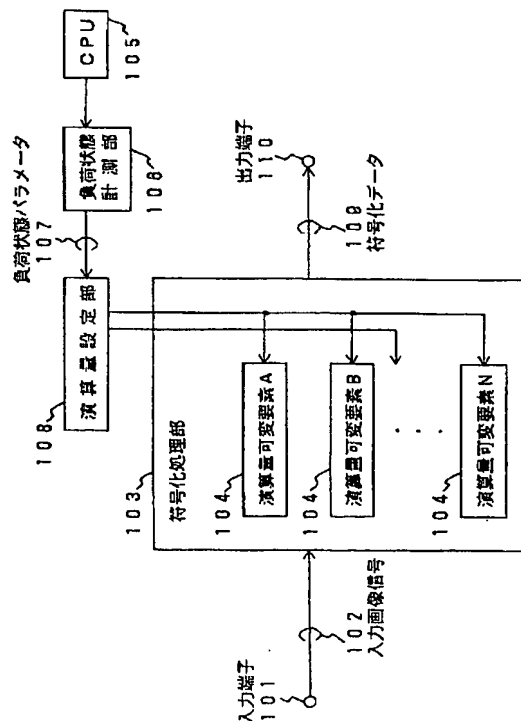
(74) 代理人 弁理士 小笠原 吉義 (外1名)

(54) 【発明の名称】 画像符号化復号化方法

(57) 【要約】

【課題】 画像信号を実時間で符号化復号化する場合に、CPUの演算能力の変化に対応して、与えられた演算能力の中で安定した高品質な符号化復号化を実現する方法を提供する。

【解決手段】 負荷状態計測部106は、CPU105の負荷状態を計測して負荷状態パラメータ107を演算量設定部108に送る。演算量設定部108は、負荷状態パラメータ107をもとに符号化処理部103の演算量を負荷状態に応じて設定する。符号化処理部103は、演算量可変要素104の演算量が設定された演算量になるような処理アルゴリズムにて入力画像信号102を符号化処理し、符号化する。



【特許請求の範囲】

【請求項1】 画像信号を符号化または復号化する場合、符号器、復号器または外付けのコンピュータのCPUの負荷状態を参照し、その負荷状態を計測することによって、符号化または復号化処理に要する演算量を負荷状態に応じて適応的に変化させることを特徴とする画像符号化復号化方法。

【請求項2】 請求項1記載の画像符号化復号化方法において、符号化または復号化処理の中に演算量を変化させることができる要素が複数個存在する時に、それぞれ10の要素の演算量を、符号器、復号器または外付けのコンピュータのCPUの負荷状態の度合によって適応的に決定することを特徴とする画像符号化復号化方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ソフトウェアにより計算機上で画像信号を実時間でかつ高品質に符号化・復号化することを目的とした画像符号化および復号化方法に関するものである。

【0002】

【従来の技術】近年のコンピュータのCPU能力の伸びに対応して、ソフトウェアによって画像の符号化および復号化を実行することがしばしば行われる。ソフトウェアによる符号化復号化は、そのハンドリングの容易性から、将来が期待されている。

【0003】ソフトウェアによる符号化の手順の例は以下になる。まず、ディスクあるいはテープ等に蓄積されている画像信号から、ある単位の画像データをコンピュータのメインメモリ上に呼び出す。呼び出し単位は、種々考えられるが、動画像信号の場合、1フレームごとに呼び出されることが多い。ある単位の画像データが呼び出されると、その画像データに対して、符号化処理が施され符号化データが出力される。メインメモリ上の画像データの符号化処理が終了した時点で、次の処理単位の画像データをディスクあるいはテープから読み出す。

【0004】復号側においても、入力符号化データとなり、出力が復号画像信号となることを除いて、符号化側と同様のプロセスで処理が実行される。このような方法によれば、動画像の呼び出し単位となっている画像データを10実時間表示した場合の時間 t_i と、その画像データを符号化処理するのに要する時間 t_c の関係が、 $t_i \geq t_c$ である場合には、実時間での処理が可能となる。たとえば、NTSC形式のテレビジョン信号に対しては、1/30秒ごとに新たなフレームが入力されるが、これに対し、1フレームの符号化処理が1/30秒以内で終了すれば良いことになる。

【0005】しかしながら、符号化処理に際して、高画質化のために動き補償等の大きな演算量を要するアルゴリズムを用いると、CPU等の演算処理能力の関係で、

この条件が成り立たなくなるおそれがある。また、演算量が比較的少ないアルゴリズムを用いる場合でも、コンピュータ上でのソフトウェア処理の場合、他のジョブとの関係で、符号化処理に割り当てることのできる演算能力が実効的に減少し、同様にこの条件が成立しなくなる場合がある。

【0006】従来は、符号化処理に割り当てる演算能力が減少した場合でも実時間での処理を可能とするために、あるフレームの処理が終了するまで、次々に入力されるフレームを読み飛ばし、処理が終了した時点で入力可能なフレームから符号化処理を開始するという手段で対処していた。

【0007】図8に、従来の符号化側の処理のブロック構成を示す。入力端子801から入力された入力画像信号802は、1フレーム分の容量を持つフレームメモリ803に蓄積され、スイッチ804を通じて符号化処理部805に入力され、符号化データ806が出力端子807に出力される。ここで、スイッチ804は符号化処理部805からのフィードバック信号によってオン/オフ20され、前フレームの画像データの符号化処理が実行されている時にはオフ、符号化処理が終了している状態ではオンになるように設定される。

【0008】

【発明が解決しようとする課題】従来の方法によれば、動画像のフレーム間隔の時間 t_i と、その画像データを符号化処理するのに要する時間 t_c の関係が $t_i \geq t_c$ である場合には、スイッチ804は必ずオンの状態になり、すべてのフレームの処理が可能となる。

【0009】しかしながら、 $t_i < t_c$ である場合には、 t_c の大きさによってスイッチ804がオフの状態になる時間があり、その時にフレームメモリ803に入力されたフレームのデータは、符号化されずに捨てられることになる。たとえば、 t_c が t_i の3倍である時には、3フレームに1フレームの割合でしか符号化が行われず、残りの2フレームは符号化されない。符号化されなかったフレームは受信側では符号化されたフレームの繰り返しなどで再現されるが、フレーム繰り返しで再現された画像は動きの再現性が極めて悪い。

【0010】一方、 t_c は一定ではないので、図9に示すようにCPUにかかっている負荷の状態により時々刻々と変化し、符号化されるフレームの間隔もそれに10じて変化するという状況が起こりうる。

【0011】このように、従来のような方法では、フレーム読み飛ばしにより実時間での符号化復号化処理を可能にしているために、計算機のCPUに負荷がかかると急激に画質が劣化し、かつ画質の変化がCPUへの負荷のかかり方の変化に非常に敏感で激しく、安定した再生画像を提供できないという問題点があった。

【0012】

【課題を解決するための手段】本発明は、上記の問題点

3

を解決するためになされたものであって、ソフトウェアによって計算機上で画像信号を符号化または復号化する際に、計算機のCPUの負荷状態を参照し、CPUの負荷状態の度合によって、符号化または復号化処理に要する演算量を適応的に変化させることを特徴とするものである。

【0013】具体的には、他のジョブなどによりCPUの負荷が増加している時には、符号化または復号化にかかる演算量を少なくするように処理アルゴリズムを切り替え、また逆に、CPUの負荷が減少している時には、符号化または復号化にかかる演算量を多くして、複雑な処理が可能になるように処理アルゴリズムを切り替える。

【0014】以下、符号化側を例にとりて説明する。図1に本発明を実施するための符号化側のブロック構成を示す。まず、入力端子101から入力された符号化対象映像の入力画像信号102が実時間で入力され、符号化処理部103へ送られる。符号化処理部103は、N個の演算量を変化させることのできるモジュール（演算量可変要素104）の集まり、すなわち、第1の演算量可変要素、第2の演算量可変要素、・・・、第Nの演算量可変要素104に分割されているとし、これらの演算を通して符号化処理が行われる。

【0015】一方、符号化処理がなされている計算機上のCPU105における負荷状態計測部106では、そのCPU105にかかっているすべての負荷の状態を計測する。計測された負荷状態パラメータ107は演算量設定部108に送られ、N個の演算量可変要素に対する演算量が設定された後、N個の演算量可変要素104へ送られ、各演算量可変要素104においては、定められた演算量になるような処理アルゴリズムにて処理が行われる。処理されて得られた符号化データ109が、出力端子110に出力される。

【0016】このような細かい演算量の制御は、従来のハードウェアによる符号化復号化処理では対処が困難であったが、ソフトウェアによる符号化復号化処理であれば、プログラム中のパラメータ変更やループの回数制限などで簡単に対応することが可能であり、実現が容易である。

【0017】以上述べたような本発明の方法によれば、計算機のCPUの負荷状態を参照しCPUの負荷状態の度合を計測することにより、他のジョブなどによってCPUの負荷が増加している時には、符号化または復号化にかかる演算量を少なくでき、また逆に、CPUの負荷が減少している時には、符号化または復号化にかかる演算量を多くして、複雑な処理が可能となる。また、演算量可変要素における演算量の変化は非常に細かく設定することが可能である。

【0018】

【発明の実施の形態】以下に本発明の第1の実施の形態

4

を示す。本実施例は、画像信号を動き補償して差分を取り、差分を量子化し可変長符号を割り当てる動き補償フレーム間符号化の符号化部を想定する。

【0019】図2に本実施例のブロック図を示す。まず、入力端子201から入力された現フレームの入力画像信号202を16×16の小ブロックに分割し、分割された小ブロックに対して、フレームメモリ203に蓄えられている過去または未来のフレームの画像データから、動きベクトル探索部204において求められる動きベクトル205に従って最も類似するブロックを求め、求められた動きベクトル205に従って、動き補償部206において過去または未来のフレームの画像データから動き補償予測データ207を得る。

【0020】現フレームの小ブロックの入力画像信号202と動き補償予測データ207を用いて、差分器208で動き補償フレーム間差分値209を算出する。動き補償フレーム間差分値209に対して、量子化部210において定められた量子化ステップ幅で量子化して量子化レベル番号を得、符号割り当て部211において量子化レベル番号に符号を割り当てて差分符号化データ212を得る。また、動きベクトル205にも符号を割り当てて、差分符号化データ212とともに多重化部213で多重し、符号化データ214として出力端子215から出力する。

【0021】一方、量子化レベル番号を、逆量子化部216にて逆量子化して量子化代表値を得た後、動き補償予測データ207と加算器217にて加算し、その結果をフレームメモリ203に書き込む。

【0022】このうち、動きベクトル探索部204での処理は、以下のようになっている。まず、現フレームの小ブロックB(i, j)に対して、フレームメモリ203中の過去または未来のフレームから空間的にB(i, j)を(u, v)だけずらした小ブロックB'(i+u, j+v)との差分絶対値和を計算する。

【0023】

【数1】

$$D(u, v) = \sum_{i=0}^{15} \sum_{j=0}^{15} |B(i, j) - B'(i+u, j+v)|$$

【0024】(u, v)を定められた領域で変化させ、D(i, j)の最小値を与えるような(u, v)を最終的な動きベクトル(U, V)として決定する。この動きベクトルの決定については、(u, v)を変化させる精度（探索精度、何画素おきに動きを探索するか）を細かくし、かつ差分絶対値和D(u, v)を算出する際にも16×16画素すべてに対して演算するようにして算出すれば、信頼性の高い動きベクトルが得られ、結果として得られるフレーム間差分値が小さくなって、符号化効率率が向上することが知られている。しかしながら、この動きベクトル探索に関しては非常に演算量が多いので、

5

特にソフトウェアにおいて計算機上で実時間で符号化処理を行なおうとした場合に最もネックとなる部分である。

【0025】ここで、 (u, v) を変化させる精度と、差分絶対値和 $D(i, j)$ を算出する際に用いる画素数については、必ずしも全てを用いなくてもある程度の信頼性の上で動きベクトルを求めることが可能である。したがって、この2つは、演算量可変要素として扱うことが可能である。

【0026】そこで、動きベクトル探索部204のうち、探索ベクトル設定部221と差分絶対値和計算部222に対して、CPU218からの演算量制御を施すようにする。計算機上のCPU218における負荷状態計測部219では、そのCPU218にかかっているすべての負荷の状態を計測する。計測された負荷状態パラメータ220は演算量設定部223に送られ、2個の演算量可変要素、すなわち探索ベクトル設定部221と差分絶対値和計算部222に対する演算量が設定される。

【0027】動きベクトル探索部204の動作を詳細に説明すれば、探索ベクトル設定部221において、演算量設定部223で定められた演算量を実現するための演算量パラメータ225に基づいて探索ベクトル（動きベクトルの候補）が設定され、各探索ベクトルに対して、ブロックシフト部224にて、フレームメモリ203に蓄積されている過去または未来の画像がシフトされ、差分絶対値和計算部222にて、現フレームの小ブロック画像データ（入力画像信号202）との差分絶対値和が計算される。この場合、差分絶対値和を計算するとき用いる画素数は、演算量設定部223で定められた演算量を実現するための演算量パラメータ225に基づいて定められる。

【0028】次に、差分絶対値和が最小値検出部226に送られ、これにより差分絶対値和が最小になるような探索ベクトルが、動きベクトル決定部227にて決定され、最終的な動きベクトル205として求められる。

【0029】探索ベクトル設定部221および差分絶対値和計算部222での具体的な演算量制御は、以下のような方法で行われる。計測されたCPU218の負荷状

$$F(u, v) = \frac{2C(u)C(v)}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) \cos\left\{\frac{(2i+1)u\pi}{2N}\right\} \cos\left\{\frac{(2j+1)v\pi}{2N}\right\}$$

【0035】で表される。 $(N=8)$ ただし、

【0036】

【数3】

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}} & w = 0 \\ 1 & w = 1, 2, 3, \dots, N-1 \end{cases}$$

【0037】である。まず、入力端子601から入力された現フレームの画像データである入力画像信号602は、離散コサイン変換部607へ送られる。その後、量

6

態パラメータ (S) 220を、それが最大の場合を0、最小の場合を1とすると、探索ベクトル設定部221における探索精度 P は、

$$P = 1/S_1$$

差分絶対値和計算部222において用いる画素数 Q は、

$$Q = 256 \times S_2$$

と定められる。ただし、 $S_1 \times S_2 = S$ である。たとえば、 $S = 0.125$ の場合には、 $S_1 = 0.25$ 、 $S_2 = 0.5$ とすることで実現できる。 $S_1 = 0.25$ の場合、 $P = 4$ となり、この意味するところは、整数単位にすべてのベクトルを探索するのではなく、探索精度を $1/4$ に減らすことである。

【0030】すなわち、図3に探索ベクトルの演算量制御の例として示すように、探索ベクトル（動きベクトルの候補となるベクトル）を4ベクトルに1つだけにする。また、差分絶対値和を計算する画素数 $Q = 128$ となるので、図4に示すように1画素おきに和をとっていくことになる。全体としての演算量は、 $S_1 \times S_2 = 0.125$ に抑えることができる。

【0031】種々の負荷状態パラメータ (S) 220に対する P 、 Q の選択方法の例を図5に示す。このようにすれば、計算機にかかっている負荷の変動に対して非常に細かく演算量を設定するアルゴリズムを提供できる。また、本例では、演算量可変要素が2種類存在するので、請求項2の発明のように、どちらの演算量可変要素をどのくらいに設定するかを全体の負荷状態によって適応的に定めることにより、より画質の高い復号画像を提供することができる。

【0032】図6は、本発明の第2の実施の形態を示すものである。本実施例では、画像信号を水平8画素、垂直8画素の小ブロックに分割し、小ブロックごとに離散コサイン変換して変換係数を量子化し、可変長符号を割り当てる場合を想定する。

【0033】2次元離散コサイン変換の変換式を以下に示す。入力された小ブロック $f(i, j)$ に対して変換係数 $F(u, v)$ は、

【0034】

【数2】

子化部608へ送られ、量子化レベル番号を得て、符号割り当て部609において量子化レベル番号に符号を割り当てて、符号化データ610として出力端子611から出力する。

【0038】このうち離散コサイン変換部607での処理は、以下のようになっている。まず、CPU603の現在の負荷状態を負荷状態計測部604により求め、得られた負荷状態パラメータ (S) 605に基づいて、演算量設定部606において演算量が設定される。設定さ

れた演算量は、離散コサイン変換部607に送られる。離散コサイン変換部607では、この設定演算量により演算量を適応的に省略する処理を行う。すなわち、負荷状態パラメータ(S)605が最大の場合を0、最小の場合を1としたとき、演算量可変要素として離散コサイン変換係数の計算個数Nを、Sによって以下のように切り替える。

【0039】 $N=8 \times 8 \times S$

このとき図7に示すように、ブロック内での周波数成分を高次の最後まで計算せずに、低次の周波数成分から順番にN個を計算して、ある閾値をそれ以降の高次の成分の演算は行わずに、強制的に0とする。強制的に0にした係数については、当然、引き続き量子化の制御も必要ない。このようにすれば、CPU603に負荷がかかっているときほど、演算量を少なくすることができ、かつ高次の係数から演算を行うようにしているので、画質への影響の少ない符号化が可能となる。

【0040】符号化の際の処理について説明したが、復号化の際にも以上説明した符号化処理に対応する処理が行われることは言うまでもない。その復号化処理の詳細については、以上の符号化の説明から容易に類推できるので説明を省略する。

【0041】

【発明の効果】以上述べたような方法によれば、符号化・復号化に実効的に使用できる演算能力を演算量にフィードバックすることが可能となるので、実時間性を保持した符号化・復号化が可能となる。また、演算量可変要素を適当に選ぶことによって、実行的に使用できる演算能力の微妙な変化にも対応が可能で、急激にフレーム脱落が起こることもないので、与えられた演算能力の中で安定した高画質な符号化を実現できる。特に、ソフトウェアによる符号化復号化処理であれば、プログラム中のパラメータ変更やループの回数制限などで簡単に実現することが可能であり、実現性も非常に高いと言える。

【図面の簡単な説明】

【図1】本発明を実施するための符号化側のブロック構成を示す図である。

【図2】本発明の第1の実施の形態のブロック構成を示す図である。

【図3】第1の実施の形態における探索ベクトルの演算量制御の例を示す図である。

【図4】第1の実施の形態における差分絶対値和計算部の演算量制御の例を示す図である。

【図5】負荷状態パラメータSに対するP、Qの選択方法の例を示す図である。

【図6】本発明の第2の実施の形態のブロック構成を示す図である。

【図7】本発明の第2の実施の形態における離散コサイン変換の計算方法を示す図である。

【図8】従来の符号化側の処理のブロック構成を示す図

である。

【図9】従来の符号化方法の問題点を示す図である。

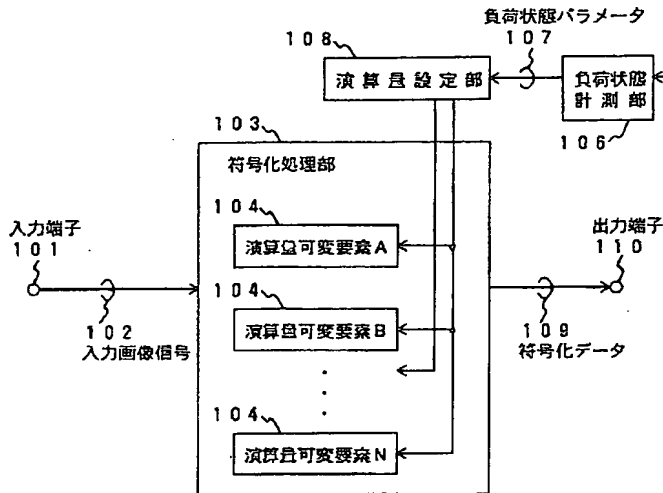
【符号の説明】

101 入力端子
102 入力画像信号
103 符号化処理部
104 演算量可変要素
105 CPU
106 負荷状態計測部
107 負荷状態パラメータ
108 演算量設定部
109 符号化データ
110 出力端子
201 入力端子
202 入力画像信号
203 フレームメモリ
204 動きベクトル探索部
205 動きベクトル
206 動き補償部
207 動き補償予測データ
208 差分器
209 動き補償フレーム間差分値
210 量子化部
211 符号割り当て部
212 差分符号化データ
213 多重化部
214 符号化データ
215 出力端子
216 逆量子化部
217 加算器
218 CPU
219 負荷状態計測部
220 負荷状態パラメータ
221 探索ベクトル設定部
222 差分絶対値和計算部
223 演算量設定部
224 ブロックシフト部
225 演算量パラメータ
226 最小値検出部
227 動きベクトル決定部
601 入力端子
602 入力画像信号
603 CPU
604 負荷状態計測部
605 負荷状態パラメータ
606 演算量設定部
607 離散コサイン変換部
608 量子化部
609 符号割り当て部
610 符号化データ

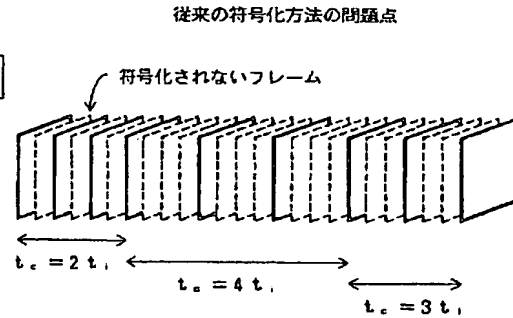
611 出力端子
801 入力端子
802 入力画像信号
803 フレームメモリ

804 スイッチ
805 符号化処理部
806 符号化データ
807 出力端子

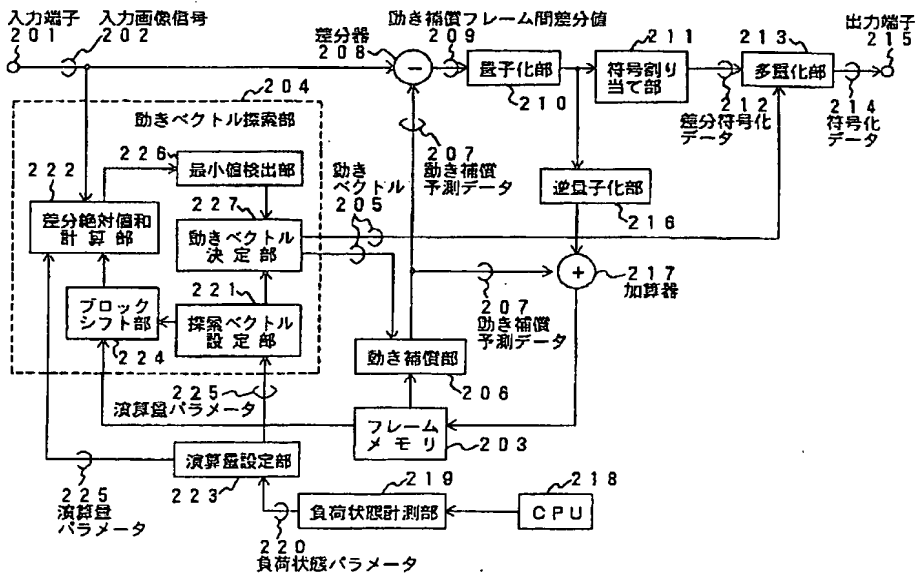
【図1】



【図9】

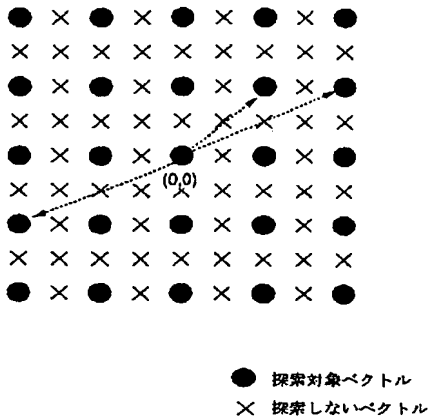


【図2】



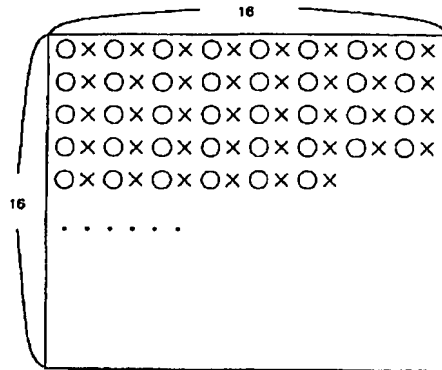
【図3】

探索ベクトルの演算量制御の例



【図4】

差分絶対値和計算部の演算量制御の例

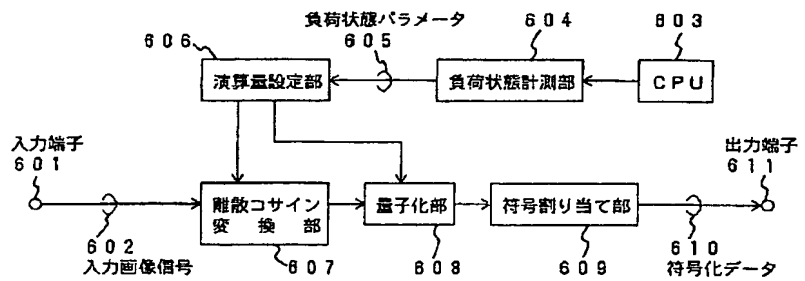


【図5】

P, Qの選択方法の例の説明図

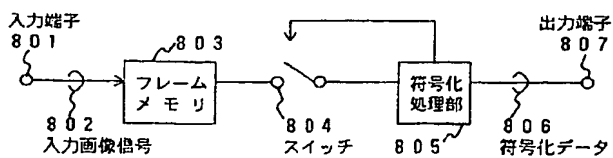
S	S ₁	S ₂
0.5	1	0.5
0.25	1	0.25
0.125	0.25	0.5
0.1	0.5	0.2
0.08	0.4	0.2
0.04	0.2	0.2

【図6】



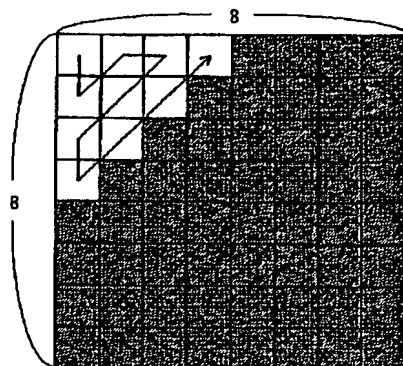

【図8】

従来の符号化方法



【図7】

第2の実施例における離散コサイン変換の計算例

 DCT係数の演算を行う DCT係数の演算を行わない

This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.